An Approach of Transforming Ontologies into Relational Databases

Loan T.T. Ho(✉), Chi P.T. Tran, and Quang Hoang

Hue University of Sciences, Ho Chi Minh City, VietNam
{thuyloan13488,phuongchi0910,hquang10}@gmail.com

Abstract. The main purpose of the Semantic Web expresses the information as intelligent forms, enables better computers and people to work in cooperation. Ontologies, as meaning providers, play a key role in this effort. Currently, the OWL Web Ontology Language is used as a description ontology language that is the main of technique for storing ontologies. On the other hand, traditional relational database systems are often used as best mechanisms for storing, querying, manipulating the information, and have some benefits such as transaction management, security. For this reason, there is a need for storing ontologies represented by OWL into relational databases is proposed. Therefore, principles of mapping OWL concepts to relational database schemas are presented with an implemented tool, which is the purpose of this paper.

Keywords: Ontology · Relational database · OWL · Mapping · Transformation

1 Introduction

Nowadays, ontology is the platform of the Semantic web and has become more popular as the model of the information in specified domain. Ontologies play an important role by providing the means to align the knowledge performs. The language used to describe the ontology which is the Web Ontology Language OWL designed by W3C. OWL can be seen as an representative schema language that can be used to provide flexible access to data. Unfortunately, interpreting of schema statements in the OWL is different from explaining of similar statements in a relational database setting. This can lead to problems in data-centric applications, where OWLs interpretation of statements intended as constraints may be confusing and/or inappropriate [1].

In addition, for sharing OWL, we often publish OWL files on the web which can be used anywhere. This can lead to problems, how must OWL files be stored in databases and efficient processing this information by user applications. For this purpose to deal with storing ontology, Relational Database (RDB) is a good candidate that have proven capabilities to handle with large amounts of data, although ontologies can be stored in various databases such as relational, object or object-relational [2]. In particular, the advantages of relational database management systems are mature, performing, robust, reliable and available.

N.T. Nguyen et al. (Eds.): ACIIDS 2015, Part I, LNAI 9011, pp. 149–158, 2015.
DOI: 10.1007/978-3-319-15702-3_15

There are some approaches of transforming ontologies into relational database that were presented in the articles [3–8]. The authors only coped with main OWL structures, and some components of OWL can not be considered in those papers. Therefore, those approaches are mainly forthright and still incomplete, or obtained relational structures are not applicable for real information systems. In this paper, we propose another approach based on above, which can be used for automatically transforming from ontologies represented by OWL into RDB schemas. The major part of concepts are mapped into relational tables, relations and attributes, other semantics of constraints and properties are stored like metadata in special tables. Using this hybrid approach, it is probable to obtain appropriate relational structures and preserve semantic information ontology.

The rest of the paper is organized as follows. Section 2 gives a brief some concepts and the detail rules of transforming from ontology constructs into relational database schemas were described. In section 3, the implementation of the proposed approach is presented. Section 4 considers the related work. Finally, we conclude overall the paper in section 5.

2 OWL Concepts and Their Mapping to RDB Concepts

2.1 OWL Class

A class, which is a basic concept of the ontology, defines a group of individuals belong together. So, the transformation of classes is the most important step to support to convert other concepts of OWL later on. A named class is mapped into one database table. As the whole ontology has an unique name of the class, and even the name of instances is unique in the class, so this table is named with the name of the class and has a primary key that is created automatically by adding *ID* as a suffix at the end, such as a *Person* class is mapped to a *Person* table that has a *PersonID* primary key.

In addition, there are class hierarchies in OWL ontology. The fundamental taxonomic construct for classes is *rdfs:subClassOf*. So that, *rdfs:subClassOf* syntax is mapped to an inheritance relationship as Fig. 1. It means that the created table corresponds to the subclass. This table gets its primary key as a foreign key that relates to its superclass table. Besides, as a class also relates other classes, so one table is created for every class in the ontology with one-to-one relations between classes and their subclasses.

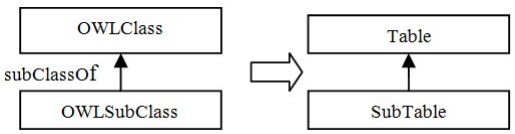

Fig. 1. Mapping OWL subclass construct to RDB

2.2 OWL Properties

In OWL, two types of properties are distinguished: Datatype properties specified a relation between instances of a class and data type values. Object properties specified a relation between instances of two classes. Furthermore, the authors in [9] said that properties can be single-valued or multivalued. If the cardinality of the property has a (maximum) value of 1, or the property is (inverse) functional, then the property is single-valued. In any other case, the property is multivalued.

Object Property. When transforming from OWL ontology into RDB schema, the object property is mapped to a foreign key or an intermediate table. Depending on the property is single-valued or multivalued, a relationship among the tables corresponded the classes can be one-to-many or many-to-many. In a case of many-to-many relation, an intermediate table must be created.

Datatype Property. In a class, for each datatype property is single-valued, it is mapped to a column in a table. This table corresponds to the class specified in the domain of the property. And the name of the column is same as the name of the datatype property. The type of the column that is specified in the range of the property is converted from XSD to SQL [9].

However, a datatype property is also multivalued, and SQL did not support multivalued columns. To deal with this problem, this property is mapped to a table. The name of the table is the datatype propertys name suffixed with *Value*. Its primary key is a combination of a corresponding column and a foreign key related to the table that corresponds to the class specified in the domain of the property. An example about a *hobby* property (i.e., a person has many hobbies). The property is mapped to a *hobbyValue* table. This table gets its primary key as a set of a *hobby* column with a varchar type and a *PersonID* foreign key that referenced to a *Person* table.

If a datatype property has a value restriction, then it is mapped to a corresponding column with a CHECK constraint. Such as a *typeOfProject* datatype property is restricted to have the same value for all instances of a *SoftwareProject* class.

```
<owl:DatatypeProperty rdf:ID="typeOfProject">
   <rdfs:domain rdf:resource="#SoftwareProject"/>
   <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty> <owl:Class rdf:ID="SoftwareProject">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#typeOfProject"/>
      <owl:hasValue rdf:resource=Software/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class> CREATE TABLE SoftwareProject
 (typeOfProject VARCHAR,
  typeOfProject CHECK (typeOfProjecy = Software));
```

In addition to the RDF datatypes, OWL provides one added construct for defining a range of data values, namely an enumerated datatype. In the case, an enumerated datatype is mapped to a column with a CHECK constraint. An example about a *sex* property in a *Person* class with a list of values *Male* and *Female*.

```
<owl:DatatypeProperty rdf:ID=sex>
 <rdfs:domain rdf:resource=#Person/>
 <rdfs:range>
  <owl:DataRange>
  <owl:oneOf>
   <rdf:List>
    <rdf:first rdf:datatype=&xsd;string>Male
   </rdf:first>
   <rdf:rest>
     <rdf:List>
        <rdf:first rdf:datatype=&xsd;string>Female
     </rdf:first>
        <rdf:rest rdf:resource=&rdf;nil/>
     </rdf:List>
   </rdf:rest>
   </rdf:List>
  </owl:oneOf>
 </owl:DataRange>
</rdfs:range> </owl:DatatypeProperty> CREATE TABLE Person (sex
VARCHAR, sex CHECK IN (Male, Female));
```

2.3 Property Characteristics

In order to provide the mechanism for enhanced reasoning about a property, OWL used property characteristics.

Symmetric Property. Properties may be stated to be symmetric. If a property is symmetric and single valued, then it is transformed to reflexive relation in RDB, i.e., the property is transformed to a foreign key. This key references to the same table that corresponds to both its domain and range class. For example, an *isSpouseOf* symmetric property (i.e., if one person is a spouse of another person, and vice versa) is mapped to an *isSpouseOf* column in a *Person* table, and this column is also a foreign key in the same table. The transformation of this property is shown as Fig. 2.

If a symmetric property is multivalued, then it is mapped to a table. This table gets its primary key as a set of foreign keys that reference to the table corresponded to both domain and range class of the property. For instance, a *hasFriend* symmetric property (i.e., if one person is a friend of many person, and vice versa) is mapped to a *Student-hasFriend* table. The transformation of this property is shown as Fig. 3.

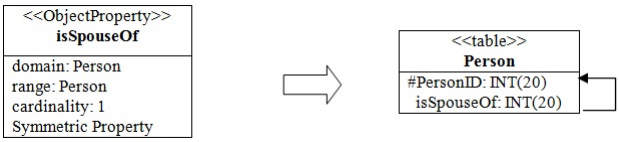

Fig. 2. An example of the symmetric property that is single-valued

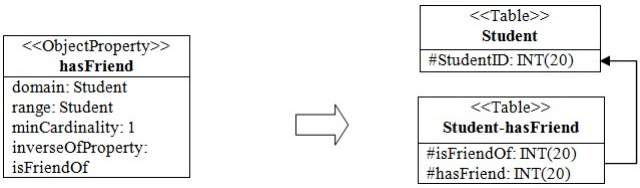

Fig. 3. An example of the symmetric property that is multivalued

Transitive Property. Properties may be stated to be transitive. The transformation of the transitive property is the same as the symmetric property. Such as a *hasAncestor* property is transitive property (i.e., if one person is an ancestor of another person, then the second person is an ancestor of the third person). The transformation of this property is shown as Fig. 4.

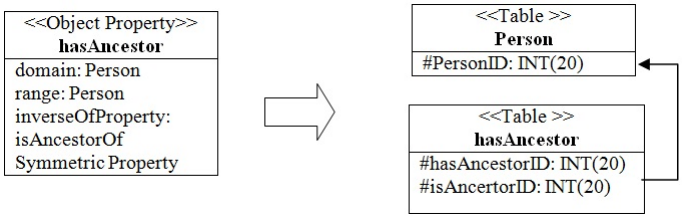

Fig. 4. An example of the transitive property that is multivalued

2.4 Property Restrictions

In OWL, we use a property restriction to constrain the range of a property in particular contexts following a variety of ways. Property restrictions can be applied both to data type properties and object properties. The context of an owl:Restriction can only be used for the various form such as the owl:allValues From, owl:someValuesFrom, owl:cardinality [10]. When transforming from OWL ontology into RDB, with aim of preserving all semantic information of ontology constraints, this information is saved in special tables like metadata tables. There are there metadata tables for each type of *AllValuesFrom*, *SomeValuesFrom*, *HasValue* restrictions and cardinality restrictions. The properties table stores the semantic of the properties in ontology. The above metadata tables are represented as follows:

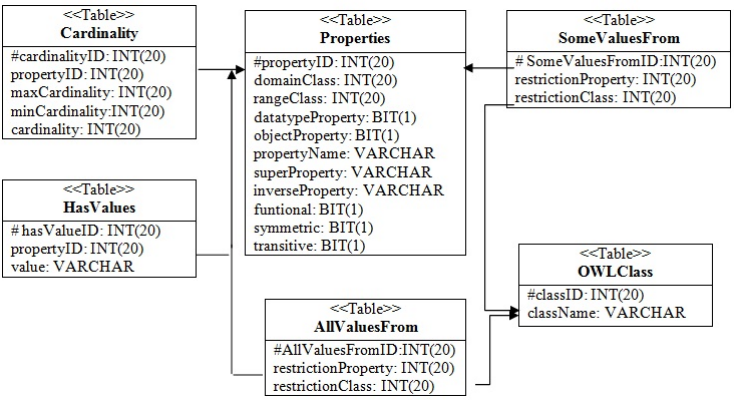

Fig. 5. Storing of OWL property restrictions in RDB

2.5 Individuals

In OWL, individuals are instances of classes. So that, after classes, properties, restriction constraints are mapped, we inserted all instances of classes into rows in the corresponding table. That is the last step of transforming from domain ontology into relational database.

3 Experiments

Based on the mapping rules, a transformation tool with java on the basis of Jena API is implemented. OWL is modeled by Protege 3.4, then we have carried out the tool. A graphical interface of the transformation tool is presented in Fig. 6. The interface shows the hierarchy of ontology classes as a tree construct, and buttons for transforming from ontology into relational database. And then, a generated text file as the output is described by SQL language. The connection between transformation tool and database sever using JDBC driver. Data are stored in My SQL 5.6 after that. We test by the UniversityOntology.owl example.

4 Related Work

In recent years, there are several approaches for addressing the issue of mapping from ontology representation in OWL to relational database [3–8]. Firstly, Kajal et al. [3] proposed a set of techniques to provide a mapping of an OWL ontology to a relational schema. The OWL2DB transformation algorithm consists of 8 steps and interprets OWL ontology like a graph. A shortcoming of that approach is only saving class, instances in RDB with other semantics represented in OWL. In 2006, based on the idea of the algorithm in [3], Lina et al. [4] developed this algorithm at a higher level, namely the transformation of constraints

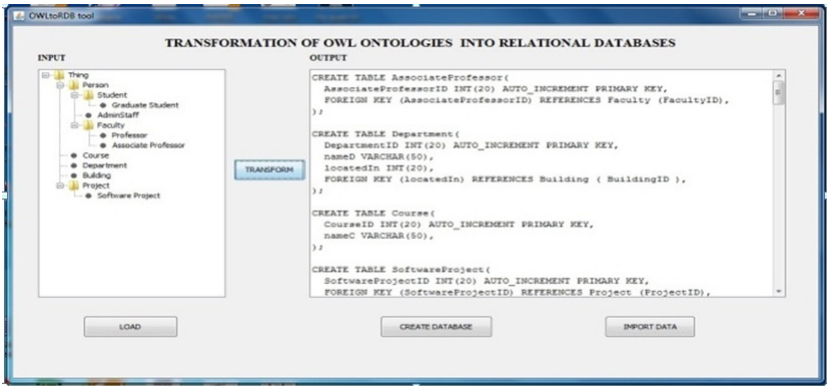

Fig. 6. The graphical interface of the transformation tool

in owl: Restriction syntax are solved. That is what the previous algorithm did not do. However, these authors said that the proposed algorithm was capable of transferring all OWL Lite and part of OWL DL syntax. Beside the direct mapping proposal, Jutas et al. [5] and Yanhui et al. [6] recommended a mediate approach for knowledge represented by ontology automatic transformation into conceptual data model, and then converted into relational database schema. The disadvantage of using ontology for conceptual data modelling is only main concepts that are transformed. In literature [7], the authors developed an ontology management system for storing data likes database management system. A single table in system called Fact Table is used for storing facts of ontology and a set of triggers which are fired when a fact is inserted, updated, or deleted from this table. However, the state of the art system is not still completed, and this transformation method does not preserve the real relational structure. On the other hand, a mechanism for generating the relational schema from a set of integrated XML files, which includes defining a set of mapping rules from the OWL ontology to the relational format, are presented by Brum et al. [8]. In Fig. 7, we summarize the above transformation proposals and tried to assemble the similar approaches in a group.

In addition, we analyzed and compared some of transformation proposals as shown in Table 1. The comparison shows that the above approaches is incomplete for mapping of OWL to relational database, just solving a part of OWL constructs. In some cases, the transformation method can be semi-automatic, e.g. they can require much user interaction. And finally, some of proposals are not implemented. Nevertheless, we realize that transformation of ontology to relational database is based on a set of rules called mapping rules, which can be extensible, because many other constructs in OWL ontology are not still considered and the direct mapping can be applicable, automatically implemented, had correctness. Therefore, this a reason why we choose a novel approach to transformation of ontologies to relational databases, which is the main contribution of this paper.

Table 1. Comparison of the proposals for transforming of OWL ontology to relational database

Transforming of RDFS/ OWL construct	Kajal [3]	Lina [4]	Jutas [5]	Yanhui [6]	Juhnyoung [7]	Brum [8]
Class	+	+	+	+	+	+
subClassOf	+	+	-	+	+	-
oneOf, dataRange	-	-	+	+	-	-
disjoinWith	-	-	-	-	-	-
equivalentClass	-	-	-	-	-	-
unionOf	-	-	-	-	-	-
complemetnOf	-	-	-	-	-	-
intersectionOf	-	-	-	-	-	-
Inviduals	+	+	+	+	-	+
Datatype property	+	+	+	+	-	+
Object property	+	+	+	+	-	+
subPropertyOf	-	+	-	-	-	-
rdfs:domain, rdfs:range	-	+	+	+	-	+
Funtional property	-	+	-	-	-	-
Transitive property	-	-	-	-	-	-
Symmetric property	-	-	-	-	-	-
inverseOf	-	-	-	-	-	-
allValuesFrom	-	+	-	-	-	-
someValuesFrom	-	+	-	-	-	-
hasValue	-	+	-	-	-	-
cardinality	-	+	-	+	-	+
maxCardinality	-	+	-	+	-	+
minCardinality	-	+	-	+	-	+
xsd:datatype	+	+	+	+	-	+
Excess	6/20	12/20	7/20	9/20	2/20	7/20
Implementation	No	Yes	Yes	Yes	Yes	Yes
Degree of automation	automatic	semi-automatic	semi-automatic	semi-automatic	semi-automatic	automatic
Extensibility	Yes	Yes	No	No	Yes	No

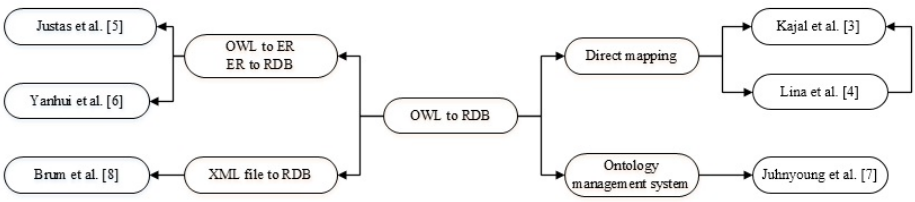

Fig. 7. The approaches of OWL ontology to relational database transformation

5 Conclusions

In this paper, we have an analysis and evaluation on some of approaches for transformation of ontology characterized as OWL into relational database.

Basing on the comparison among them, we have remarked that these proposals is incomplete. From the result of this, the combination of mapping rules is proposed in this paper. Currently, our approach is capable to transform the most of OWL DL concepts. We consider that ontology classes should be mapped to relational tables, properties are mapped to relations or attributes, instances corresponds to rows in the table. Semantic information about property restrictions and properties as symmetric, transitive, inverse functional are stored in tables like metadata. Using both direct mapping and metadata, we achieve applicable relational structure with preserving data and restrict losing the ontological constructs. A prototype tool of mapping rules which have OWL documents like the inputs and text files like the outputs, was implemented as an independent software.

However, some constructs are lost when transforming. For example, our method does not save the ontological semantics as complement class, intersection class, enumerated class. The constructs as subproperties, equivement property that are not corresponsive in RDB should be considered. In further, we intend to set up the transformation tool as a plug-in for a popular ontology editor Protege. We also extend this approach to forthcoming OWL2 that extends with current OWL.

References

1. Ian, H., Boris, M., Ulrike, S.: Bridging the gap between OWL and relational database. Journal of Web Semantics: Science, Services and Agents on the World Wide Web **7**(2), 74–89 (2009)
2. Lina, A., Christine, P., Stefano, S.: Reasoning with large ontologies stored in relational databases the OntoMinD approach. Data and Knowledge Engineering **69**(11), 1158–1180 (2010)
3. Gali, A., Chen, C.X., Claypool, K.T., Uceda-Sosa, R.: From ontology to relational databases. In: Wang, S., Tanaka, K., Zhou, S., Ling, T.-W., Guan, J., Yang, D., Grandi, F., Mangina, E.E., Song, I.-Y., Mayr, H.C. (eds.) ER Workshops 2004. LNCS, vol. 3289, pp. 278–289. Springer, Heidelberg (2004)
4. Lina, N., Ernestas, V.: Transforming ontology representation form OWL to relational database. Information Technology And Control Kaunas Technology **35**(3), 333–343 (2006)
5. Justas, T., Olegas, V.: A graph oriented model for ontology transfoemation into conceptual data model. Information Technology and Control **36**(1A), 126–132 (2007)
6. Yahui, L., Chong, X.: An ontology-based approach to build conceptual data model. In: 9th International Conference on, Sichuan (2012)

7. Juhnyoung, L., Richard, G.: Ontology management for large-scale enterprise systems. Electronic Commerce Research and Applications **5**, 2–15 (2006)
8. Brum, S.D., de Campos, A., Piveta, E.K.: Mapping OWL ontologies to relational schemas. In: IEEE International Conference, Las Vegas (2011)
9. Irina, A., Nahum, K., Ahto, K.: Storing OWL ontologies in SQL relational database. Interational Journal of Electrical, Computer and Systems Engineering **1**(4), 242–257 (2007)
10. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-features/